

Dynamic Index Coding for Wireless Broadcast Networks

Michael J. Neely , Arash Saber Tehrani , Zhen Zhang

Abstract—We consider a wireless broadcast station that transmits packets to multiple users. The packet requests for each user may overlap, and some users may already have certain packets. This presents a problem of broadcasting in the presence of side information, and is a generalization of the well known (and unsolved) index coding problem of information theory. Rather than achieving the full capacity region, we develop a *code-constrained capacity region*, which restricts attention to a pre-specified set of coding actions. We develop a dynamic max-weight algorithm that allows for random packet arrivals and supports any traffic inside the code-constrained capacity region. Further, we provide a simple set of codes based on cycles in the underlying demand graph. We show these codes are optimal for a class of broadcast relay problems.

I. INTRODUCTION

Consider a wireless broadcast station that transmits packets to N wireless users. Packets randomly arrive to the broadcast station. Each packet p is desired by one or more users in the set $\{1, \dots, N\}$. Further, there may be one or more users that already have the packet stored in their cache. The broadcast station must efficiently transmit all packets to their desired users. We assume time is slotted with unit slots $t \in \{0, 1, 2, \dots\}$, and that a single packet can be transmitted by the broadcast station on every slot. This packet is received error-free at all users. We assume that only the broadcast station can transmit, so that users cannot transmit to each other.

If the broadcast station has P packets at time 0, and no more packets arrive, then the mission can easily be completed in P slots by transmitting the packets one at a time. However, this approach ignores the side-information available at each user. Indeed, it is often possible to complete the mission in fewer than P slots if packets are allowed to be *mixed* before transmission. A simple and well known example for 2 users is the following: Suppose user 1 has packet B but wants packet A , while user 2 has packet A but wants packet B . Sending each packet individually would take 2 slots, but these demands can be met in just one slot by transmitting the mixed packet $A+B$, the bit-wise XOR of A and B . Such examples are introduced in [1][2][3] in the context of *wireless network coding*.

The general problem, where each packet is contained as side information in an arbitrary subset of the N users, is much more complex. This problem is introduced by Birk and Kol in [4][5], and is known as the *index coding problem*. Methods for completing a general index coding mission in

minimum time are unknown. However, the recent work [6] shows that if one restricts to a class of linear codes, then the minimum time is equal to the minimum rank of a certain matrix completion problem. The matrix completion problem is NP-hard in general, and hence index coding is complex even when restricted to a simpler class of codes.

Nevertheless, it is important to develop systematic approaches to these problems. That is because current wireless cellular systems cannot handle the huge traffic demands that are expected in the near future. This is largely due to the consistent growth of wireless video traffic. Fortunately, much of the traffic is for *popular content*. That is, users often download the same information. Thus, it is quite likely that a system of N users will have many instances of side information, where some users already have packets that others want. This naturally creates an index coding situation. Thus, index coding is both rich in its mathematical complexity and crucial for supporting future wireless traffic.

The problem we consider in this paper is even more complex because packets can arrive randomly over time. This is a practical scenario and creates the need for a *dynamic* approach to index coding. We assume there are M *traffic types*, where a type is defined by the subset of users that *desire* the packets and the subset that *already has* the packets. Let λ_m be the arrival rate, in packets/slot, for type M traffic. We approach this problem by restricting coding actions to an abstract set \mathcal{A} . We then show how to achieve the *code constrained capacity region* $\Lambda_{\mathcal{A}}$, being the set of all rate vectors $(\lambda_m)_{m=1}^M$ that can be supported using coding actions in the set \mathcal{A} . The set $\Lambda_{\mathcal{A}}$ is typically a strict subset of the *capacity region* Λ , which does not restrict the type of coding action. Our work can be applied to any set \mathcal{A} , and hence can be used in conjunction with any desired codes. However, we focus attention on a simple class of codes that involve only bit-wise XOR operations, based on cycles in the underlying demand graph. In special cases of *broadcast relay problems*, we show that these codes can achieve the full capacity region Λ .

The capacity region Λ is directly related to the conceptually simpler *static* problem of clearing a fixed batch of packets in minimum time. Further, index coding concepts are most easily developed in terms of the static problem. Thus, this paper is divided into two parts: We first introduce the index coding problem in the static case, and we describe example coding actions in that case. Section III extends to the dynamic case and develops two max-weight index coding techniques, one that requires knowledge of the arrival rates (λ_m) , and one that does not. The algorithms here are general and can also be used in other types of networks where controllers make sequences of actions, each action taking a different number of slots and delivering a different vector of packets.

The authors are with the Electrical Engineering department at the University of Southern California, Los Angeles, CA.

This material is supported in part by one or more of the following: the DARPA IT-MANET program grant W911NF-07-0028, the NSF Career grant CCF-0747525, NSF grant 0964479, the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory W911NF-09-2-0053.

While the static index coding problem has been studied before [6][5][4], our work provides new insight even in the static case. We introduce a new directed bipartite demand graph that allows for arbitrary demand subsets and possibly “multiple multicast” situations, where some packets are desired by more than one user. We also form a useful *weighted compressed graph* that facilitates the solution to the minimum clearance time problem in certain cases. This extends the graph models in [6], which do not consider the possibility of multiple multicast sessions. Work in [6] develops a maximum acyclic subgraph bound on minimum clearance time for problems without multiple multicast sessions. We extend this bound to our general problem using a different proof technique. Further, we consider a class of *broadcast relay problems* for which the bound can be achieved with equality.

The next section introduces index coding in the static case, shows its relation to a bipartite demand graph, and presents the acyclic subgraph bound. Section III introduces the general dynamic formulation and develops our max-weight algorithms. Section IV considers an important class of broadcast relay networks for which a simple set of codes are optimal.

II. THE STATIC MINIMUM CLEARANCE TIME PROBLEM

This section introduces the index coding problem in the static case, where we want to clear a fixed batch of packets in minimum time. Consider a wireless system with N users, P packets, and a single broadcast station. We assume N and P are positive integers. Let \mathcal{N} and \mathcal{P} represent the set of users and packets, respectively:

$$\mathcal{N} = \{1, \dots, N\}, \quad \mathcal{P} = \{1, \dots, P\}$$

The broadcast station has all packets in the set \mathcal{P} . Each user $n \in \mathcal{N}$ has an arbitrary subset of packets $\mathcal{H}_n \subseteq \mathcal{P}$, and wants to receive an arbitrary subset of packets $\mathcal{R}_n \subseteq \mathcal{P}$, where $\mathcal{H}_n \cap \mathcal{R}_n = \emptyset$, where \emptyset represents the empty set. Assume that all packets consist of B bits, all packets are independent of each other, and the B -bit binary string for each packet is uniformly distributed over each of the 2^B possibilities.

We can represent this system by a *directed bipartite demand graph* \mathcal{G} defined as follows (see Fig. 1):

- User nodes \mathcal{N} are on the left.
- Packet nodes \mathcal{P} are on the right.
- A directed link (n, p) from a user node $n \in \mathcal{N}$ to a packet node $p \in \mathcal{P}$ exists if and only if user n has packet p . That is, if and only if $p \in \mathcal{H}_n$.
- A directed link (p, n) from a packet node $p \in \mathcal{P}$ to a user node $n \in \mathcal{N}$ exists if and only if user n wants to receive packet p . That is, if and only if $p \in \mathcal{R}_n$.

As an example for the 3-user, 5-packet graph of Fig. 1, the *have* and *receive* sets for nodes 1 and 2 are:

$$\mathcal{H}_1 = \{5\}, \quad \mathcal{R}_1 = \{1, 2\}$$

$$\mathcal{H}_2 = \emptyset, \quad \mathcal{R}_2 = \{1, 2, 4\}$$

We restrict attention to packets that at least one node wants. Thus, without loss of generality, throughout we assume the

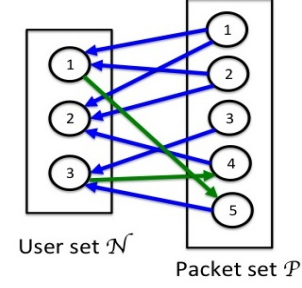


Fig. 1. An example directed bipartite demand graph with 3 users and 5 packets.

graph \mathcal{G} is such that all packet nodes $p \in \mathcal{P}$ contain at least one outgoing link. Thus:

$$\mathcal{P} = \{1, \dots, P\} = \cup_{n=1}^N \mathcal{R}_n \quad (1)$$

In this static problem, the broadcast station has all packets in the set \mathcal{P} at time 0, and no more packets ever arrive. Every slot $t \in \{0, 1, 2, \dots\}$ the broadcast station can transmit one B -bit message over the broadcast channel. This message is received without error at all of the user nodes in the set \mathcal{N} . The goal is for the broadcast station to send messages until all nodes receive the packets they desire.

Define a *mission-completing coding action* with T slots to be a sequence of messages that the broadcast station transmits over the course of T slots, such that all users are able to decode their desired packets at the end of the T slots. We restrict attention to deterministic zero-error codes that enable decoding with probability 1. The initial information held by each user $n \in \mathcal{N}$ is given by the set of packets \mathcal{H}_n (possibly empty). Let $\mathcal{M} \triangleq \{\mathcal{M}_1, \dots, \mathcal{M}_T\}$ represent the messages transmitted by the broadcast station over the course of the T slot coding action. At the end of this action, each node $n \in \mathcal{N}$ has information $\{\mathcal{H}_n, \mathcal{M}\}$. Because the coding action is assumed to complete the mission, this information is enough for each node n to decode its desired packets \mathcal{R}_n . That is, we can write:

$$\{\mathcal{H}_n, \mathcal{M}\} \iff \{\mathcal{H}_n, \mathcal{M}, \mathcal{R}_n\} \quad (2)$$

where the above represents *equivalence in the information set*, meaning that the information on the left-hand-side can be perfectly reconstructed from the information on the right-hand-side, and vice versa. Clearly the information on the left in (2) is a subset of the information on the right, and hence can trivially be reconstructed. The information on the right in (2) can be reconstructed from that on the left because the code is mission-completing.

For a given graph \mathcal{G} with P packet nodes, define $T_{\min}(\mathcal{G})$ as the *minimum clearance time* of the graph, being the minimum number of slots required to complete the mission, considering all possible coding techniques. Clearly $T_{\min}(\mathcal{G}) \leq P$. Our goal is to understand $T_{\min}(\mathcal{G})$.

For a directed graph, we say that a *simple directed cycle of length K* is a sequence of nodes $\{n_1, n_2, \dots, n_K, n_1\}$ such that (n_i, n_{i+1}) is a link in the graph for all $i \in \{1, \dots, K-1\}$, (n_K, n_1) is a link in the graph, and all nodes $\{n_1, \dots, n_K\}$ involved in the cycle are distinct. For simplicity, throughout

this paper we use the term *cycle* to represent a simple directed cycle. We say that the graph \mathcal{G} is *acyclic* if it contains no cycles. Note that directed acyclic graphs have a much different structure than undirected acyclic graphs. Indeed, the graph in Fig. 1 is acyclic even though its undirected counterpart (formed by replacing all directed links with undirected links) has cycles.

Our first result is to prove that if a directed bipartite demand graph \mathcal{G} is acyclic, then coding cannot reduce the minimum clearance time. This result was first proven in [6] in the case without “multiple multicasts,” so that each packet is desired by at most one user. That result uses an argument based on machinery of the mutual information function. It also treats a more general case where codes can have errors. Further, their proof is developed as a consequence of a more general and more complex result. Our work restricts to zero-error codes, but allows the possibility of multiple-multicast sessions. We also use a different proof technique which emphasizes the logical consequences of users being able to decode their information. Our proof uses only the following two facts:

Fact 1: Every directed acyclic graph with a finite number of nodes has at least one node with no outgoing links. Such a node is called a “leaf” node.

Fact 2: If the graph contains only one user node, then $T_{\min}(\mathcal{G}) = P$, where P is the number of packets that this user desires.

Fact 1 follows simply by starting at any node in the graph and traversing a path from node to node, using any outgoing link, until we find a leaf node (such a path cannot continue forever because the graph is finite and has no cycles). Fact 2 is a basic information theory observation about the capacity of a single error-free link.

Theorem 1: If the graph \mathcal{G} is acyclic, then $T_{\min}(\mathcal{G}) = P$, where P is the total number of packets in the graph.

Proof: See Appendix A. \square

As an example, because the graph \mathcal{G} in Fig. 1 is acyclic, we have $T_{\min}(\mathcal{G}) = 5$. Theorem 1 shows that coding cannot help if \mathcal{G} is acyclic, so that the best one can do is just transmit all packets one at a time. Therefore, any type of coding must exploit cycles on the demand graph.

A. Lower Bounds from Acyclic Subgraphs

Theorem 1 provides a simple lower bound on $T_{\min}(\mathcal{G})$ for any graph \mathcal{G} . Consider a graph \mathcal{G} , and form a subgraph \mathcal{G}' by performing one or more of the following *pruning operations*:

- Remove a packet node, and all of its incoming and outgoing links.
- Remove a user node, and all of its incoming and outgoing links.
- Remove a packet-to-user link (p, n) .

After performing these operations, we must also delete any residual packets that have no outgoing links. Any sequence of messages that completes the mission for the original graph \mathcal{G} will *also* complete the mission for the subgraph \mathcal{G}' . This leads to the following simple lemma.

Lemma 1: For any subgraph \mathcal{G}' formed from a graph \mathcal{G} by one or more of the above pruning operations, we have:

$$T_{\min}(\mathcal{G}') \leq T_{\min}(\mathcal{G})$$

Combining this lemma with Theorem 1, we see that we can take a general graph \mathcal{G} with cycles, and then perform the above pruning operations to reduce to an acyclic subgraph \mathcal{G}' . Then $T_{\min}(\mathcal{G})$ is lower bounded by the number of packets in this subgraph. Thus, the best lower bound corresponds to the acyclic subgraph generated from the above operations, and that has the largest number of remaining packets. Note that the above pruning operations do not include the removal of a user-to-packet link (n, p) (without removing either the entire user or the entire packet), because such links represent side information that can be helpful to the mission.

B. Particular code actions

Because the general index coding problem is difficult, it is useful to restrict the solution space to consider only sequences of simple types of coding actions. Recall that coding actions must exploit cycles. One natural action is the following: Suppose we have a cycle in \mathcal{G} that involves a subset of K users. For simplicity label the users $\{1, \dots, K\}$. In the cycle, user 2 wants to receive a packet X_1 that user 1 has, user 3 wants to receive a packet X_2 that user 2 has, and so on. Finally, user 1 wants to receive a packet X_K that user K has. The structure can be represented by:

$$1 \rightarrow 2 \rightarrow 3 \rightarrow \dots \rightarrow K \rightarrow 1 \quad (3)$$

where an arrow from one user to another means the left user has a packet the right user wants. Of course, the users in this cycle may want many other packets, but we are restricting attention only to the packets X_1, \dots, X_K . Assume these packets are all distinct.

In such a case, we can satisfy all K users in the cycle with the following $K - 1$ transmissions: For each $k \in \{1, \dots, K - 1\}$, the broadcast station transmits a message $\mathcal{M}_k \triangleq X_k + X_{k+1}$, where addition represents the mod-2 summation of the bits in the packets. Each user $k \in \{2, \dots, K\}$ receives its desired information by adding \mathcal{M}_{k-1} to its side information:

$$X_k + \mathcal{M}_{k-1} = X_k + (X_{k-1} + X_k) = X_{k-1}$$

Finally, user 1 performs the following computation (using the fact that it already has packet X_1):

$$\begin{aligned} & X_1 + \mathcal{M}_1 + \mathcal{M}_2 + \dots + \mathcal{M}_{K-1} \\ &= X_1 + (X_1 + X_2) + (X_2 + X_3) + \dots + (X_{K-1} + X_K) \\ &= (X_1 + X_1) + (X_2 + X_2) + \dots + (X_{K-1} + X_{K-1}) \\ &\quad + X_K \\ &= X_K \end{aligned}$$

Thus, such an operation can deliver K packets in only $K - 1$ transmissions. We call such an action a *K-cycle coding action*. We define a *1-cycle coding action* to be a direct transmission. Note that 2-cycle coding actions are the most “efficient,” having a packet/transmission efficiency ratio of $2/1$, compared to $K/(K - 1)$ for $K \geq 2$, which approaches 1 (the efficiency

of a direct transmission) as $K \rightarrow \infty$. While it is generally sub-optimal to restrict to such cyclic coding actions, doing so can still provide significant gains in comparison to direct transmission. Further, we show in Section IV that such actions are optimal for certain classes of broadcast relay problems.

Another important type of code action takes advantage of “double-cycles” in \mathcal{G} : Suppose for example that user 1 wants packet A and has packets B and C , user 2 wants packet B and has packets A and C , and user 3 wants packet C and has packets A and B . Then these demands can be fulfilled with the single transmission $A + B + C$, being a binary XOR of packets A, B, C . The efficiency ratio of this action is $3/1$.

III. DYNAMIC INDEX CODING

Now consider a dynamic setting where the broadcast station randomly receives packets from M traffic flows. Each flow $m \in \{1, \dots, M\}$ contains fixed-length packets that must be delivered to a subset \mathcal{N}_m of the users, and these packets are contained as side-information in a subset \mathcal{S}_m of the users. We assume $\mathcal{N}_m \cap \mathcal{S}_m = \emptyset$, since any user $n \in \mathcal{N}_m$ who wants the packet clearly does not already have the packet as side information. In the general case, M can be the number of all possible disjoint subset pair combinations. However, typically the value of M will be much smaller than this, such as when each traffic flow represents packets from a very large file, and there are only M active file requests.

Assume time is slotted with unit slots $t \in \{0, 1, 2, \dots\}$, and let $\mathbf{A}(t) = (A_1(t), \dots, A_M(t))$ be the number of packets that arrive from each flow on slot t . For simplicity of exposition, we assume the vector $\mathbf{A}(t)$ is i.i.d. over slots with expectation:

$$\mathbb{E}\{\mathbf{A}(t)\} = \boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_M)$$

where λ_m is the arrival rate of packets from flow m , in units of packets/slot. For simplicity, we assume that $A_m(t) \in \{0, 1\}$ for all m and all t , so that at most one new packet can arrive per flow per slot. This is reasonable because the maximum delivery rate in the system is one packet per slot, and so any packets that arrive as a burst can be “smoothed” and delivered to the network layer at the broadcast station one slot at a time. Packets of each flow m are stored in a separate queue kept at the broadcast station, and exit the queue upon delivery to their intended users.

We now segment the timeline into frames, each frame consisting of an integer number of slots. At the beginning of each frame r , the network controller chooses a coding action $\alpha[r]$ within an abstract set \mathcal{A} of possible actions. For each $\alpha \in \mathcal{A}$, there is a *frame size* $T(\alpha)$ and a *clearance vector* $\boldsymbol{\mu}(\alpha)$. The frame size $T(\alpha)$ is the number of slots required to implement action α , and is assumed to be a positive integer. The clearance vector $\boldsymbol{\mu}(\alpha)$ has components $(\mu_1(\alpha), \dots, \mu_M(\alpha))$, where $\mu_m(\alpha)$ is the number of type m packets delivered as a result of action α . We assume $\mu_m(\alpha)$ is a non-negative integer. When frame r ends, a new frame starts and the controller chooses a (possibly new) action $\alpha[r+1] \in \mathcal{A}$. We assume each coding action only uses packets that are delivered as a result of that action, so that there is no “partial information” that can be exploited on future frames.

We further assume there are a finite (but arbitrarily large) number of coding actions in the set \mathcal{A} , and that there are positive numbers T_{max} and μ_{max} such that $1 \leq T(\alpha) \leq T_{max}$ and $0 \leq \mu(\alpha) \leq \mu_{max}$ for all $\alpha \in \mathcal{A}$.

Assume that frame 0 starts at time 0. Define $t[0] = 0$, and for $r \in \{0, 1, 2, \dots\}$ define $t[r]$ as the slot that starts frame r . Let $\mathbf{Q}[r] = (Q_1[r], \dots, Q_M[r])$ be the queue backlog vector at the beginning of each frame $r \in \{0, 1, 2, \dots\}$. Then:

$$Q_m[r+1] = \max[Q_m[r] - \mu_m(\alpha[r]), 0] + arrivals_m[r] \quad (4)$$

where $arrivals_m[r]$ is the number of type m arrivals during frame r :

$$arrivals_m[r] \triangleq \sum_{\tau=t[r]}^{t[r]+T(\alpha[r])-1} A_m(\tau) \quad (5)$$

The $\max[\cdot, 0]$ operator in the queue update equation (4) in principle allows actions $\alpha[r] \in \mathcal{A}$ to be chosen independently of the queue backlog at the beginning of a frame. In this case, if the action $\alpha[r]$ attempts to deliver one or more packets from queues that are empty, *null* packets are created and delivered. In practice, these null packets do not need to be delivered.

Our focus is on index coding problems with action sets \mathcal{A} defined by a specific set coding options, such as the set of all cyclic coding actions. For example, an action α that is a 2-cyclic coding action that uses packets of type m and k has $T(\alpha) = 1$ and $\boldsymbol{\mu}(\alpha)$ being a binary vector with 1s in entries m and k and zeros elsewhere. However, the above model is general and can also apply to other types of problems, such as multi-hop networks where actions $\alpha \in \mathcal{A}$ represent some sequence of multi-hop network coding.

A. The Code-Constrained Capacity Region

We say that queue $Q_m[r]$ is *rate stable* if:

$$\lim_{R \rightarrow \infty} \frac{Q_m[R]}{R} = 0 \quad (\text{with probability } 1)$$

It is not difficult to show that $Q_m[R]$ is rate stable if and only if the arrival rate λ_m is equal to the delivery rate of type m traffic [7]. The *code-constrained capacity region* $\Lambda_{\mathcal{A}}$ is the set of all rate vectors $(\lambda_1, \dots, \lambda_M)$ for which there exists an algorithm for selecting $\alpha[r] \in \mathcal{A}$ over frames that makes all queues rate stable.

Theorem 2: A rate vector $\boldsymbol{\lambda}$ is in the code-constrained capacity region $\Lambda_{\mathcal{A}}$ if and only if there exist probabilities $p(\alpha)$ such that $\sum_{\alpha \in \mathcal{A}} p(\alpha) = 1$ and:

$$\lambda_m \leq \frac{\sum_{\alpha \in \mathcal{A}} p(\alpha) \mu_m(\alpha)}{\sum_{\alpha \in \mathcal{A}} p(\alpha) T(\alpha)} \quad \forall m \in \{1, \dots, M\} \quad (6)$$

Proof: The proof that such probabilities $p(\alpha)$ necessarily exist whenever $\boldsymbol{\lambda} \in \Lambda_{\mathcal{A}}$ is given in Appendix B. Below we prove sufficiency. Suppose such probabilities $p(\alpha)$ exist that satisfy (6). We want to show that $\boldsymbol{\lambda} \in \Lambda_{\mathcal{A}}$. To do so, we design an algorithm that makes all queues $Q_m[r]$ in (4) rate stable. By rate stability theory in [7], it suffices to design an algorithm that has a frame average arrival rate to each queue $Q_m[r]$ that is less than or equal to the frame average service rate (both in units of packets/frame).

Consider the algorithm that, every frame r , independently chooses action $\alpha \in \mathcal{A}$ with probability $p(\alpha)$. Let $\alpha^*[r]$ represent this random action chosen on frame r . Then $\{T(\alpha^*[r])\}_{r=0}^\infty$ is an i.i.d. sequence, as is $\{\mu_m(\alpha^*[r])\}_{r=0}^\infty$ for each $m \in \{1, \dots, M\}$. By the law of large numbers, the frame average arrival rate $\overline{arrivals}_m$ and the frame average service $\bar{\mu}_m$ (both in packets/frame) are equal to the following with probability 1:

$$\begin{aligned}\bar{\mu}_m &= \mathbb{E}\{\mu_m(\alpha^*[r])\} = \sum_{\alpha \in \mathcal{A}} p(\alpha) \mu_m(\alpha) \\ \overline{arrivals}_m &= \lambda_m \mathbb{E}\{T(\alpha^*[r])\} = \lambda_m \sum_{\alpha \in \mathcal{A}} p(\alpha) T(\alpha)\end{aligned}$$

We thus have for each $m \in \{1, \dots, M\}$:

$$\frac{\overline{arrivals}_m}{\bar{\mu}_m} = \frac{\lambda_m \sum_{\alpha \in \mathcal{A}} p(\alpha) T(\alpha)}{\sum_{\alpha \in \mathcal{A}} p(\alpha) \mu_m(\alpha)} \leq 1$$

where the final inequality follows by (6). \square

B. Max-Weight Queueing Protocols

Theorem 2 shows that all traffic can be supported by a *stationary and randomized* algorithm that independently chooses actions $\alpha^*[r] \in \mathcal{A}$ with probability distribution $p(\alpha)$. This does not require knowledge of the queue backlogs. However, computing probabilities $p(\alpha)$ that satisfy (6) would require knowledge of the arrival rates λ_m , and is a difficult computational task even if these rates are known. We provide two *dynamic* algorithms that use queue backlog information. These can also be viewed as online computation algorithms for computing probabilities $p(\alpha)$. Both are similar in spirit to the max-weight approach to dynamic scheduling in [8], but the variable frame lengths require a non-trivial extended analysis. Our first algorithm assumes knowledge of the arrival rates λ_m .

Max-Weight Code Selection Algorithm 1 (Known λ): At the beginning of each frame r , observe the queue backlogs $Q_m[r]$ and perform the following:

- Choose code action $\alpha[r] \in \mathcal{A}$ as the maximizer of:

$$\sum_{m=1}^M Q_m[r] [\mu_m(\alpha[r]) - \lambda_m T(\alpha[r])] \quad (7)$$

where ties are broken arbitrarily.

- Update the queue equation via (4).

The next algorithm uses a ratio rule, and does not require knowledge of the rates λ_m :

Max-Weight Code Selection Algorithm 2 (Unknown λ): At the beginning of each frame r , observe the queue backlogs $Q_m[r]$ and perform the following:

- Choose code action $\alpha[r] \in \mathcal{A}$ as the maximizer of:

$$\sum_{m=1}^M Q_m[r] \left[\frac{\mu_m(\alpha[r])}{T(\alpha[r])} \right] \quad (8)$$

where ties are broken arbitrarily.

- Update the queue equation via (4).

Theorem 3: Suppose that $\lambda \in \Lambda_{\mathcal{A}}$. Then all queues are rate stable under either of the two algorithms above.

Proof: See Appendix C. \square

It can further be shown that if there is a value ρ such that $0 \leq \rho < 1$, and if $\lambda \in \rho \Lambda_{\mathcal{A}}$, being a ρ -scaled version of $\Lambda_{\mathcal{A}}$,

then both algorithms give average queue size $O(1/(1-\rho))$. Thus, the average backlog bound increases to infinity as the arrival rates are pushed closer to the boundary of the capacity region. This is proven in Appendix D.

Define $\tilde{\mathcal{A}}$ as the action space that restricts to direct transmissions, 2-cycle code actions, 3-cycle code actions, and the 1-slot $A+B+C$ code action that exploits double cycles, as described in Section II-B. Algorithm 2 has a particularly simple implementation on action space $\tilde{\mathcal{A}}$ and when each packet has at most one destination. Indeed, we note that cycles can be defined purely on the user set \mathcal{N} , and any candidate cycle that involves a user-to-user part $i \rightarrow j$ should use a packet of commodity $m \in \{1, \dots, M\}$ that maximizes $Q_m[r]$ over all commodities m that consist of packets intended for user j and contained as side information at user i .

C. Example Simulation for 3 Users

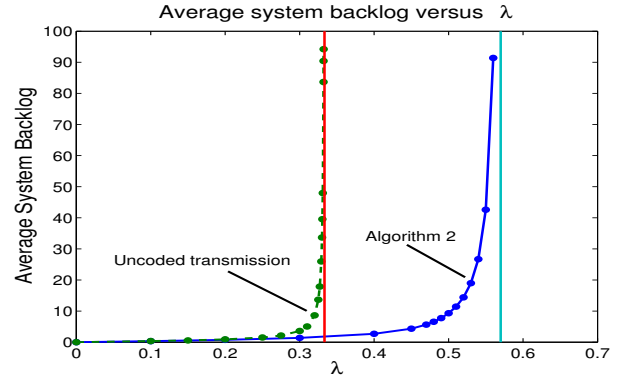


Fig. 2. Simulation of dynamic index coding for a 3 user system.

Fig. 2 presents simulation results for a system with $N = 3$ users, with action space $\tilde{\mathcal{A}}$ as defined above. We consider only algorithm 2, which does not require knowledge of rates λ_m , and compare against uncoded transmissions. All packets are intended for at most one user. Packets intended for user $n \in \{1, 2, 3\}$ arrive as independent Bernoulli processes with identical rates λ . We assume each packet is independently in the cache of the other two users with probability 1/2. Thus, there are four types of packets intended for user 1: Packets not contained as side information anywhere, packets contained as side information at user 2 only, packets contained as side information at user 3 only, and packets contained as side information at both users 2 and 3. Users 2 and 3 similarly have 4 traffic types, for a total of $M = 12$ traffic types.

Each data point in Fig. 2 represents a simulation over 5 million frames at a given value of λ . The figure plots the resulting total average number of packets in the system (summed over all 12 queues). The case of direct (uncoded) transmission is also shown. Uncoded transmission can support a maximum rate of $\lambda = 1/3$ (for a total traffic rate of 1). It is seen that algorithm 2 can significantly outperform uncoded transmission, achieving stability at rates up to $\lambda = 0.57$ (for a total traffic rate of 1.71).

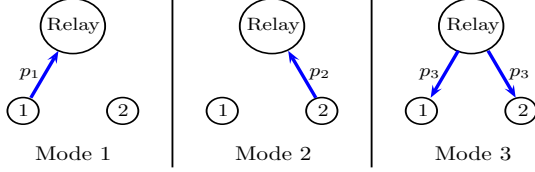


Fig. 3. An illustration of a 2-user broadcast relay system, with the 3 possible transmission modes shown. In mode 3, packet p_3 is received at both users.

IV. BROADCAST RELAY NETWORKS

Consider now the following related problem: There are again N users and a single broadcast station. However, the broadcast station initially has no information, and acts as a relay to transfer independent unicast data between the users. Further, the users only know their own data, and initially have no knowledge of data sourced at other users. Time is again slotted, and every slot we can choose from one of $N+1$ modes of transmission. The first N transmission modes involve an error-free packet transmission from a single user to the relay. The $(N+1)$ th transmission mode is where the relay broadcasts a single packet that is received error-free at each of the N users. Fig. 3 illustrates an example system with 2 users, where the 3 possible transmission modes are shown. For simplicity, we assume the user transmissions cannot be overheard by other users, and the users first send all packets to the relay. The relay then can make coding decisions for its downlink transmissions.

A. The Minimum Clearance Time Relay Problem

First consider a static problem where a batch of packets must be delivered in minimum time. Let P_{ij} represent the number of packets that user i wants to send to user j , where $i, j \in \{1, \dots, N\}$. All packets are independent, and the total number of packets is P , where:

$$P = \sum_{i=1}^N \sum_{j=1}^N P_{ij}$$

This problem is related to the index coding problem as follows: Suppose on the first P slots, all users send their packets to the relay on the uplink channels. It remains for the relay to send all users the desired data, and these users have side information. The resulting side information graph \mathcal{G} is the same as in the general index coding problem. However, it has the following *special structure*: The only user that has side information about a packet is the source user of the packet. Specifically:

- Each packet is contained as side information in exactly one user. Thus, each packet node of \mathcal{G} has a single incoming link from some user that is its source.
- Each packet has exactly one user as its destination. Thus, each packet node of \mathcal{G} has a single outgoing link to some user that is its destination.

This special structure leads to a simplified graphical model for demands, which we call the *weighted compressed graph* $\mathcal{WC}(\mathcal{G})$ of \mathcal{G} . The graph $\mathcal{WC}(\mathcal{G})$ is formed from \mathcal{G} as follows: It is a directed graph defined on the user nodes \mathcal{N} only, and contains a link (a, b) if and only if the original graph \mathcal{G} specifies that user node a has a packet that user node b wants.

Further, each link (a, b) is given a positive integer weight P_{ab} , the number of packets user a wants to send to user b . It is easy to show that $\mathcal{WC}(\mathcal{G})$ is acyclic if and only if \mathcal{G} is acyclic (see Appendix E). Hence, coding can only help if $\mathcal{WC}(\mathcal{G})$ contains cycles, and so $T_{\min}(\mathcal{G}) = P$ whenever $\mathcal{WC}(\mathcal{G})$ is acyclic.

We say the weighted compressed graph $\mathcal{WC}(\mathcal{G})$ has *disjoint cycles* if each link participates in at most one simple cycle. An example is shown in Fig. 4. Consider such a graph that has C disjoint cycles. Let w_c^{\min} be the min-weight link on each disjoint cycle $c \in \{1, \dots, C\}$.

Theorem 4: If the broadcast relay problem has a weighted compressed graph $\mathcal{WC}(\mathcal{G})$ with disjoint cycles, then:

$$T_{\min}(\mathcal{G}) = P - \sum_{c=1}^C w_c^{\min} \quad (9)$$

and so the full clearance time (including the P uplink transmissions) is the above number plus P . Further, optimality can be achieved over the class of cyclic coding actions, as described in Section II-B.

As an example, the graph $\mathcal{WC}(\mathcal{G})$ in Fig. 4a has $P = 48$, three disjoint cycles with $w_1^{\min} = 4$, $w_2^{\min} = 4$, $w_3^{\min} = 1$, and so $T_{\min}(\mathcal{G}) = 48 - 4 - 4 - 1 = 39$.

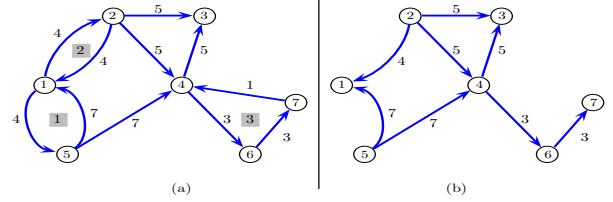


Fig. 4. (a) A graph $\mathcal{WC}(\mathcal{G})$ with three disjoint cycles, and (b) its pruned graph $\mathcal{WC}(\mathcal{G}')$.

Proof: First prune the graph $\mathcal{WC}(\mathcal{G})$ by removing the min-weight link on each of the disjoint cycles (breaking ties arbitrarily). This corresponds to removing those packets from the original graph \mathcal{G} , to produce a new graph \mathcal{G}' with exactly $P - \sum_{c=1}^C w_c^{\min}$ packets. The weighted compressed graph $\mathcal{WC}(\mathcal{G}')$ is the subgraph of $\mathcal{WC}(\mathcal{G})$ with the min-weight links on each disjoint cycle removed (see Fig. 4a and Fig. 4b). Both \mathcal{G}' and $\mathcal{WC}(\mathcal{G}')$ are acyclic, and so:

$$T_{\min}(\mathcal{G}) \geq T_{\min}(\mathcal{G}') = P - \sum_{c=1}^C w_c^{\min}$$

It remains only to construct a coding algorithm that achieves this lower bound. This can be done easily by using w_c^{\min} separate cyclic coding actions for each of the disjoint cycles (using a k -cycle coding action for any cycle of length k), and then directly transmitting the remaining packets. \square

B. Traffic Structure and Optimality of Cyclic Coding

Suppose we have a broadcast relay problem with N users, packet matrix (P_{ij}) , and with the following additional structure: Each user $i \in \{1, \dots, N\}$ wants to send data to only one other user. That is, the matrix (P_{ij}) has at most one non-zero entry in each row $i \in \{1, \dots, N\}$. We now show that the resulting graph $\mathcal{WC}(\mathcal{G})$ has disjoint cycles. To see this, suppose it is not true, so that there are two overlapping cycles. Then there must be a shared link (a, b) that continues to a link

(b, k) for cycle 1 and (b, m) for cycle 2, where $k \neq m$. This means node b has two outgoing links, a contradiction because matrix (P_{ij}) has at most one non-zero entry in row b , and hence at most one outgoing link from node b . We conclude that $\mathcal{WC}(\mathcal{G})$ has disjoint cycles, and so cyclic coding is optimal via Theorem 4.

A similar argument holds if each user wants to *receive* from at most one other user, so that (P_{ij}) has at most one non-zero entry in every column. Again, $\mathcal{WC}(\mathcal{G})$ has disjoint cycles, and so cyclic coding is optimal.

C. Dynamic Broadcast Relay Scheduling

Now consider the dynamic case where packets from source user i and destination user j arrive with rate λ_{ij} packets/slot. Suppose we have an abstract set of coding actions \mathcal{A} , where each action involves a subset of packets, and first transmits these packets to the relay before any coding at the relay. Let $T(\alpha)$ be the number of slots to complete the action, and $(\mu_{ij}(\alpha))$ be the matrix of packets delivered by the action. It can be shown that capacity can be approached arbitrarily closely by repetitions of minimum-clearance time scheduling on large blocks of the incoming data (similar to the capacity treatment in [9] for a limit of large packet size). Hence, if $T_{min}(\mathcal{G})$ can be optimally solved using only cyclic-coding actions, then capacity is also achieved in the max-weight algorithms when \mathcal{A} is restricted to cyclic-coding actions. It follows that such actions are optimal for rate matrices (λ_{ij}) with at most one non-zero entry per row, and for rate matrices (λ_{ij}) with at most one non-zero entry per column.

D. Counterexamples

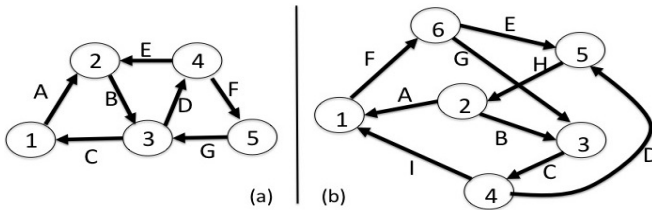


Fig. 5. Two example graphs $\mathcal{WC}(\mathcal{G})$ for broadcast relay problems.

Can we minimize clearance time by grabbing any available 2-cycle, then any available 3-cycle if no 2-cycle is available, and so on? Not necessarily. A simple counterexample is shown in Fig. 5a. The graph has 5 users and 7 packets $\{A, \dots, G\}$, where each link has a single packet. Using the middle 3-cycle $2 \rightarrow 3 \rightarrow 4 \rightarrow 2$ by transmitting $B + D$ and $D + E$ leaves a remaining acyclic graph with 4 packets, and hence would take 4 more transmissions, for a total of 6 slots. However, using the two side cycles (with 2 transmissions each) and then transmitting the remaining packet E clears everything in 5 slots, which is optimal because the maximum acyclic subgraph has 5 packets (just remove links B and D).

One may wonder if all broadcast relay graphs can be optimally cleared with cyclic coding. We can show this is true for $N = 2$ and $N = 3$ (see Appendix E and F). However, this is not true in general for $N > 3$. Fig. 5b shows a counterexample

with $N = 6$. Suppose each link has a single packet, so that we have 9 packets $\{A, \dots, I\}$. It can be shown that the maximum acyclic subgraph has 7 packets, and so $T_{min}(\mathcal{G}) \geq 7$, but the best cyclic coding method uses 8 slots. Here is a way to achieve 7 slots: Send messages $M_1 = E + G + F$, $M_2 = H + E$, $M_3 = H + D$, $M_4 = A + B + H$, $M_5 = C + B$, $M_6 = C + G$, $M_7 = C + I + D$. The decodings at users 2, 3, 4, 5, 6 are straightforward by combining their side information with just a single message. The decoding at user 1 is done as follows: $M_1 + M_2 + M_3 + M_6 + M_7 = F + I$. Since user 1 knows F , it can decode I . $M_3 + M_4 + M_5 + M_7 = A + I$, since user 1 knows I it can get A .

V. CONCLUSIONS

This work presents a dynamic approach to index coding. This problem is important for future wireless communication, where there may be many instances of side information that can be exploited. While optimal index coding for general problems seems to be intractable, we develop a code-constrained capacity region, which restricts actions to a pre-specified set of codes. Two max-weight algorithms were developed that can support randomly arriving traffic whenever the arrival rate vector is inside the code-constrained capacity region. The first algorithm requires knowledge of the rate vector, and the second does not. Simulations verify network stability up to the boundary of the code-constrained capacity region, and illustrate improvements in both throughput and delay over uncoded transmission. For coding to provide gains in comparison to direct transmission, it must exploit cycles in the demand graph. A simple set of codes based on cycles was considered and shown to be optimal (so that the code constrained capacity region is equal to the unconstrained capacity region) for certain classes of broadcast relay networks. These results add to the theory of information networks, and can be used to improve efficiency in wireless communication systems.

APPENDIX A — PROOF OF THEOREM 1

Proof: (Theorem 1) We already know that $T_{min}(\mathcal{G}) \leq P$. It suffices to show that $T_{min}(\mathcal{G}) \geq P$. Consider any mission-completing coding action that takes T slots. We show that $T \geq P$. Let \mathcal{M} be the sequence of messages transmitted. Then every node $n \in \mathcal{N}$ is able to decode its desired packets, being packets in the set \mathcal{R}_n , from the information $\{\mathcal{H}_n, \mathcal{M}\}$, being the information it has at the end of the coding action. That is, we have:

$$\{\mathcal{H}_n, \mathcal{M}\} \iff \{\mathcal{H}_n, \mathcal{M}, \mathcal{R}_n\} \quad \forall n \in \{1, \dots, N\} \quad (10)$$

Because the graph is acyclic, there must be at least one node with no outgoing links (by Fact 1). Choose such a node, and label this node n_1 . The node n_1 cannot be a packet node, because we have assumed that all packet nodes have outgoing links. Thus, $n_1 \in \mathcal{N}$. Because node n_1 has no outgoing links, it has $\mathcal{H}_{n_1} = \emptyset$ and thus has no initial side information about any of the packets. Thus, it is able to decode all packets in the set \mathcal{R}_{n_1} by the messages \mathcal{M} alone. That is:

$$\mathcal{M} \iff \{\mathcal{M}, \mathcal{R}_{n_1}\} \quad (11)$$

We want to show that this node n_1 can decode *all* packets in the set \mathcal{P} , so that:

$$\mathcal{M} \iff \{\mathcal{M}, \mathcal{P}\} \quad (12)$$

If we can show that (12) holds, then the sequence of messages \mathcal{M} is also sufficient to deliver P independent packets to node n_1 , and node n_1 did not have any initial side information about these packets. Thus, the number of slots T used in the coding action must be at least P by Fact 2, proving the result. Thus, it suffices to prove (12).

We prove (12) by induction on k , for $k \in \{1, \dots, N-1\}$: Assume that there is a labeling of k distinct user nodes $\{n_1, n_2, \dots, n_k\}$ such that:

$$\{\mathcal{M}\} \iff \{\mathcal{M}, \mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_k}\} \quad (13)$$

This property holds for the base case $k=1$ by (11). We now assume that (13) holds for a general $k \in \{1, \dots, N-1\}$, and prove it must also hold for $k+1$. Take the graph \mathcal{G} , and delete the user nodes $\{n_1, \dots, n_k\}$, also deleting all links outgoing from and incoming to these nodes. This may create packet nodes with no outgoing links: Delete all such packet nodes. Note that all deleted packet nodes (if any) must be in the set $\{\mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_k}\}$, being the set of packets desired by the users that are deleted. The resulting subgraph must still be acyclic, and hence it must have a node n_{k+1} with no outgoing links. This node must be a user node, as we have deleted all packet nodes with no outgoing links.

Because the user node n_{k+1} has no outgoing links, it either had $\mathcal{H}_{n_{k+1}} = \phi$ (so that it never had any outgoing links), or all of its outgoing links were pointing to packet nodes that we have deleted, and so those packets were in the set $\{\mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_k}\}$. That is, we must have $\mathcal{H}_{n_{k+1}} \subseteq \{\mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_k}\}$. Therefore:

$$\{\mathcal{M}, \mathcal{H}_{n_{k+1}}\} \subseteq \{\mathcal{M}, \mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_k}\} \quad (14)$$

However, at the end of the coding action, node n_{k+1} has exactly the information on the left-hand-side of (14), and hence this information is sufficient to decode all packets in the set $\mathcal{R}_{n_{k+1}}$. Thus, the information on the right-hand-side of (14) must also be sufficient to decode $\mathcal{R}_{n_{k+1}}$, so that:

$$\{\mathcal{M}, \mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_k}\} \iff \{\mathcal{M}, \mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_k}, \mathcal{R}_{n_{k+1}}\}$$

But this together with (13) yields:

$$\{\mathcal{M}\} \iff \{\mathcal{M}, \mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_k}, \mathcal{R}_{n_{k+1}}\}$$

which completes the induction step.

By induction over $k \in \{1, \dots, N-1\}$, it follows that:

$$\{\mathcal{M}\} \iff \{\mathcal{M}, \mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_N}\} \quad (15)$$

However, by re-labeling we have:

$$\{\mathcal{R}_{n_1}, \dots, \mathcal{R}_{n_N}\} = \{\mathcal{R}_1, \dots, \mathcal{R}_N\} = \mathcal{P} \quad (16)$$

where the final equality holds by (1). Combining (15) and (16) proves (12). \square

APPENDIX B – PROOF OF NECESSITY FOR THEOREM 2

Let $\{\alpha[r]\}_{r=0}^\infty$ be a sequence of actions, chosen over frames, that makes all queues $Q_m[r]$ rate stable. We show there must exist probabilities $p(\alpha)$ that satisfy (6). For each positive integer R and each $m \in \{1, \dots, M\}$, define $\bar{a}_m[R]$ and $\bar{\mu}_m[R]$ as the following averages over the first R frames:

$$\begin{aligned} \bar{a}_m[R] &\triangleq \frac{1}{R} \sum_{r=0}^{R-1} arrivals_m[r] \\ \bar{\mu}_m[R] &\triangleq \frac{1}{R} \sum_{r=0}^{R-1} \mu_m(\alpha[r]) \end{aligned}$$

where $arrivals_m[r]$ is defined in (5). Now define $\mathcal{F}(\alpha, R)$ as the set of frames $r \in \{0, \dots, R-1\}$ that use action α , and define $|\mathcal{F}(\alpha, R)|$ as the number of these frames, so that $\sum_{\alpha \in \mathcal{A}} |\mathcal{F}(\alpha, R)| = R$. We then have:

$$\bar{a}_m[R] = \sum_{\alpha \in \mathcal{A}} \frac{|\mathcal{F}(\alpha, R)|}{R} \times \frac{1}{|\mathcal{F}(\alpha, R)|} \sum_{r \in \mathcal{F}(\alpha, R)} arrivals_m[r] \quad (17)$$

$$\bar{\mu}_m[R] = \sum_{\alpha \in \mathcal{A}} \frac{|\mathcal{F}(\alpha, R)|}{R} \mu_m(\alpha) \quad (18)$$

The set \mathcal{A} is finite. Thus, the values $\{(|\mathcal{F}(\alpha, R)|/R)\}_{R=1}^\infty$ can be viewed as an infinite sequence of bounded vectors (with dimension equal to the size of set \mathcal{A}) defined on the index $R \in \{1, 2, 3, \dots\}$, and hence must have a convergent subsequence. Let R_k represent the sequence of frames on this subsequence, so that there are values $p(\alpha)$ for all $\alpha \in \mathcal{A}$ such that:

$$\lim_{k \rightarrow \infty} |\mathcal{F}(\alpha, R_k)|/R_k = p(\alpha)$$

Further, by (18) we have for all $m \in \{1, \dots, M\}$:

$$\lim_{k \rightarrow \infty} \bar{\mu}_m[R_k] = \sum_{\alpha \in \mathcal{A}} p(\alpha) \mu_m(\alpha) \quad (19)$$

Likewise, from (17) and the law of large numbers (used over each $\alpha \in \mathcal{A}$ for which $\lim_{k \rightarrow \infty} |\mathcal{F}(\alpha, R_k)| = \infty$, and noting that $arrivals_m[r]$ is i.i.d. with mean $T(\alpha)\lambda_m$ for all $r \in \mathcal{F}(\alpha, R)$) we have with probability 1:

$$\lim_{k \rightarrow \infty} \bar{a}_m[R_k] = \sum_{\alpha \in \mathcal{A}} p(\alpha) T(\alpha) \lambda_m \quad (20)$$

Because $|\mathcal{F}(\alpha, R_k)|/R_k \geq 0$ for all $\alpha \in \mathcal{A}$ and all R_k , and $\sum_{\alpha \in \mathcal{A}} |\mathcal{F}(\alpha, R_k)|/R_k = 1$ for all R_k , the same holds for the limiting values $p(\alpha)$. That is, $p(\alpha) \geq 0$ for all $\alpha \in \mathcal{A}$, and: $\sum_{\alpha \in \mathcal{A}} p(\alpha) = 1$. Because each queue $Q_m[r]$ is rate stable, we have with probability 1 that for all $m \in \{1, \dots, M\}$:

$$\lim_{k \rightarrow \infty} \frac{Q_m[R_k]}{R_k} = 0 \quad (21)$$

However, from the queue update equation (4) we have for all $r \in \{0, 1, 2, \dots\}$:

$$Q_m[r+1] \geq Q_m[r] - \mu_m(\alpha[r]) + arrivals_m[r]$$

Summing the above over $r \in \{0, 1, \dots, R_k-1\}$ and dividing by R_k yields:

$$\frac{Q_m[R_k] - Q_m[0]}{R_k} \geq -\bar{\mu}_m[R_k] + \bar{a}_m[R_k]$$

Taking a limit as $k \rightarrow \infty$ and using (19)-(21) yields:

$$0 \geq -\sum_{\alpha \in \mathcal{A}} p(\alpha) \mu_m(\alpha) + \lambda_m \sum_{\alpha \in \mathcal{A}} p(\alpha) T(\alpha) \quad (22)$$

This proves the result.

APPENDIX C — PROOF OF THEOREM 3

We first prove rate stability for Algorithm 2, which uses a ratio rule. The proof for Algorithm 1 is simpler and is given after. We have the following preliminary lemma.

Lemma 2: (Sufficient Condition for Rate Stability [10]): Let $Q[r]$ be a non-negative stochastic process defined over the integers $r \in \{0, 1, 2, \dots\}$. Suppose there are constants B, C, D such that for all frames $r \in \{0, 1, 2, \dots\}$ we have:

$$\mathbb{E} \{ (Q[r+1] - Q[r])^2 \} \leq D \quad (23)$$

$$\mathbb{E} \{ Q[r]^2 \} \leq Br + C \quad (24)$$

Then $\lim_{r \rightarrow \infty} Q[r]/r = 0$ with probability 1.

The condition (23) is immediately satisfied in our system because the queue changes over any frame are bounded. Thus, to prove rate stability, it suffices to show that (24) holds for all queues and all frames. That is, it suffices to prove the second moment of queue backlog grows at most linearly.

For each frame $r \in \{0, 1, 2, \dots\}$, define the following quadratic function $L[r]$, called a *Lyapunov function*:

$$L[r] \triangleq \frac{1}{2} \sum_{m=1}^M Q_m[r]^2$$

Define the *conditional Lyapunov drift* $\Delta[r]$ to be the expected change in $L[r]$ from one frame to the next:

$$\Delta[r] \triangleq \mathbb{E} \{ L[r+1] - L[r] | \mathbf{Q}[r] \}$$

where $\mathbf{Q}[r] = (Q_1[r], \dots, Q_M[r])$ is the queue backlog vector on frame r . The above conditional expectation is with respect to the random arrivals over the frame and the (possibly random) coding action chosen for the frame.

Lemma 3: Under any (possibly randomized) decision for $\alpha[r] \in \mathcal{A}$ that is *causal* (i.e., that does not know the future values of arrivals over the frame), we have for each frame r :

$$\Delta[r] \leq B + \sum_{m=1}^M Q_m[r] \mathbb{E} \{ \lambda_m T(\alpha[r]) - \mu_m(\alpha[r]) | \mathbf{Q}[r] \}$$

where B is a finite constant that satisfies:

$$B \geq \frac{1}{2} \sum_{m=1}^M \mathbb{E} \{ \text{arrivals}_m[r]^2 + \mu_m(\alpha[r])^2 | \mathbf{Q}[r] \}$$

Such a finite constant B exists because frame sizes are bounded, as are the arrivals per slot.

Proof: For simplicity of notation, define $b_m[r] \triangleq \mu_m(\alpha[r])$, and $a_m[r] \triangleq \text{arrivals}_m[r]$. The queue update equation is thus:

$$Q_m[r+1] = \max[Q_m[r] - b_m[r], 0] + a_m[r]$$

Note that for any non-negative values Q, a, b we have:

$$(\max[Q - b, 0] + a)^2 \leq Q^2 + b^2 + a^2 + 2Q(a - b)$$

Using this and squaring the queue update equation yields:

$$Q_m[r+1]^2 \leq Q_m[r]^2 + b_m[r]^2 + a_m[r]^2 + 2Q_m[r](a_m[r] - b_m[r])$$

Summing over all m , dividing by 2, and taking conditional expectations yields:

$$\Delta[r] \leq B + \sum_{m=1}^M Q_m[r] \mathbb{E} \{ a_m[r] - b_m[r] | \mathbf{Q}[r] \} \quad (25)$$

Now note that:¹

$$\begin{aligned} \mathbb{E} \{ a_m[r] | \mathbf{Q}[r] \} &= \mathbb{E} \left\{ \sum_{\tau=t[r]}^{t[r]+T(\alpha[r])-1} A_m(\tau) | \mathbf{Q}[r] \right\} \\ &= \mathbb{E} \{ \lambda_m T[r] | \mathbf{Q}[r] \} \end{aligned} \quad (26)$$

Plugging this identity into (25) proves the result. \square

We now prove that Algorithm 2 yields rate stability.

Proof: (Theorem 3—Stability Under Algorithm 2) Suppose that Algorithm 2 is used, so that we choose $\alpha[r]$ every frame r via (8). We first claim that for each frame r and for all possible $\mathbf{Q}[r]$ we have:

$$\begin{aligned} \frac{\mathbb{E} \left\{ \sum_{m=1}^M Q_m[r] \mu_m(\alpha[r]) | \mathbf{Q}[r] \right\}}{\mathbb{E} \{ T(\alpha[r]) | \mathbf{Q}[r] \}} &\geq \\ \frac{\mathbb{E} \left\{ \sum_{m=1}^M Q_m[r] \mu_m(\alpha^*[r]) | \mathbf{Q}[r] \right\}}{\mathbb{E} \{ T(\alpha^*[r]) | \mathbf{Q}[r] \}} &\end{aligned} \quad (27)$$

where $\alpha^*[r]$ is any other (possibly randomized) code action that could be chosen over the options in the set \mathcal{A} . This can be shown as follows: Suppose we want to choose $\alpha[r] \in \mathcal{A}$ via a possibly randomized decision, to maximize the ratio of expectations in the left-hand-side of (27). Such a decision would satisfy (27) by definition, since it would maximize the ratio of expectations over all alternative policies $\alpha^*[r]$. However, it is known that such a maximum is achieved via a *pure policy* that chooses a particular $\alpha \in \mathcal{A}$ with probability 1 (see Chapter 7 of [7]). The best pure policy is thus the one that observes the queue backlogs $\mathbf{Q}[r]$ and chooses $\alpha[r] \in \mathcal{A}$ to maximize the deterministic ratio, which is exactly how Algorithm 2 chooses its action (see (8)).

Thus, (27) holds. We can rewrite (27) as:

$$\begin{aligned} \frac{\mathbb{E} \left\{ \sum_{m=1}^M Q_m[r] \mu_m(\alpha[r]) | \mathbf{Q}[r] \right\}}{\mathbb{E} \{ T(\alpha[r]) | \mathbf{Q}[r] \}} &\geq \\ \sum_{m=1}^M Q_m[r] \frac{\mathbb{E} \{ \mu_m(\alpha^*[r]) | \mathbf{Q}[r] \}}{\mathbb{E} \{ T(\alpha^*[r]) | \mathbf{Q}[r] \}} &\end{aligned} \quad (28)$$

We can thus plug any alternative (possibly randomized) decision $\alpha^*[r]$ into the right-hand-side of (28). Consider the randomized algorithm that independently selects $\alpha \in \mathcal{A}$ every frame, independent of queue backlogs, according to the distribution $p(\alpha)$ in Theorem 2. Let $\alpha^*[r]$ represent the randomized decision under this policy. Then from (6) we have for all $m \in \{1, \dots, M\}$:

$$\lambda_m \leq \frac{\mathbb{E} \{ \mu_m(\alpha^*[r]) \}}{\mathbb{E} \{ T(\alpha^*[r]) \}} = \frac{\mathbb{E} \{ \mu_m(\alpha^*[r]) | \mathbf{Q}[r] \}}{\mathbb{E} \{ T(\alpha^*[r]) | \mathbf{Q}[r] \}} \quad (29)$$

¹Equality (26) uses causality and the i.i.d. nature of the arrival process. It is formally proven by conditioning on $T(\alpha[r])$ and using iterated expectations.

where the last equality holds because $\alpha^*[r]$ is chosen independently of $\mathbf{Q}[r]$. Using this in (28) yields:

$$\frac{\mathbb{E} \left\{ \sum_{m=1}^M Q_m[r] \mu_m(\alpha[r]) | \mathbf{Q}[r] \right\}}{\mathbb{E} \{T(\alpha[r]) | \mathbf{Q}[r]\}} \geq \sum_{m=1}^M Q_m[r] \lambda_m$$

Rearranging terms above yields:

$$\sum_{m=1}^M Q_m[r] \mathbb{E} \{ \lambda_m T(\alpha[r]) - \mu_m(\alpha[r]) | \mathbf{Q}[r] \} \leq 0 \quad (30)$$

Plugging (30) into the drift bound of Lemma 3 yields:

$$\Delta[r] \leq B$$

Taking expectations of the above and using the definition of $\Delta[r]$ yields:

$$\mathbb{E} \{L[r+1]\} - \mathbb{E} \{L[r]\} \leq B \quad \forall r \in \{0, 1, 2, \dots\}$$

Summing the above over $r \in \{0, 1, \dots, R-1\}$ yields:

$$\mathbb{E} \{L[R]\} - \mathbb{E} \{L[0]\} \leq BR$$

and hence for all $R > 0$:

$$\sum_{m=1}^M \mathbb{E} \{Q_m[R]^2\} \leq 2\mathbb{E} \{L[0]\} + 2BR$$

Thus, the second moments of all queues grow at most linearly, from which we guarantee rate stability by Lemma 2. \square

We now prove that Algorithm 1 yields rate stability.

Proof: (Theorem 3—Stability Under Algorithm 1) Note that Algorithm 1 is designed to observe queue backlogs $\mathbf{Q}[r]$ every frame r , and take a control action $\alpha[r] \in \mathcal{A}$ to minimize the right-hand-side of the drift bound in Lemma 3. Therefore, we have:

$$\Delta[r] \leq B + \sum_{m=1}^M Q_m[r] \mathbb{E} \{ \lambda_m T(\alpha^*[r]) - \mu_m(\alpha^*[r]) | \mathbf{Q}[r] \}$$

where $\alpha^*[r]$ is any other (possibly randomized) decision. If $\alpha^*[r]$ makes a decision independent of $\mathbf{Q}[r]$ we have:

$$\Delta[r] \leq B + \sum_{m=1}^M Q_m[r] \mathbb{E} \{ \lambda_m T(\alpha^*[r]) - \mu_m(\alpha^*[r]) \} \quad (31)$$

Consider again randomized algorithm $\alpha^*[r]$ that independently and randomly selects an action in \mathcal{A} every frame, independent of queue backlogs, according to the distribution $p(\alpha)$ in Theorem 2. Then (29) again holds, so that for all $m \in \{1, \dots, M\}$:

$$\mathbb{E} \{ \lambda_m T(\alpha^*[r]) - \mu_m(\alpha^*[r]) \} \leq 0$$

Substituting the above into the right-hand-side of (31) gives:

$$\Delta[r] \leq B$$

from which we then obtain rate stability in the same way as in the proof for Algorithm 2. \square

APPENDIX D — PROOF OF THE QUEUE SIZE BOUND

Here we show that if $\lambda \in \rho \Lambda_{\mathcal{A}}$, where $0 \leq \rho < 1$, then both Algorithm 1 and Algorithm 2 yield finite average backlog of size $O(1/(1-\rho))$.

Proof: (Queue Bound for Algorithm 1) Because $\lambda \in \rho \Lambda_{\mathcal{A}}$, we have:

$$(\lambda_m/\rho) \in \Lambda_{\mathcal{A}}$$

Thus, from Theorem 2 there is a randomized algorithm $\alpha^*[r]$ that makes decisions independent of queue backlogs to yield the following for all $m \in \{1, \dots, M\}$:

$$\frac{\lambda_m}{\rho} \leq \frac{\mathbb{E} \{ \mu_m(\alpha^*[r]) \}}{\mathbb{E} \{ T(\alpha^*[r]) \}} \quad (32)$$

Define $T^* \triangleq \mathbb{E} \{ T(\alpha^*[r]) \}$. Using this and rearranging the above gives:

$$\mathbb{E} \{ \mu_m(\alpha^*[r]) \} \geq \lambda_m T^* / \rho \quad \forall m \in \{1, \dots, M\}$$

Substituting the above into (31) gives:

$$\Delta[r] \leq B + \sum_{m=1}^M Q_m[r] T^* \lambda_m (1 - 1/\rho)$$

Taking expectations and using the definition of $\Delta[r]$ gives:

$$\mathbb{E} \{L[r+1]\} - \mathbb{E} \{L[r]\} \leq B + \sum_{m=1}^M \mathbb{E} \{Q_m[r]\} T^* \lambda_m (1 - 1/\rho)$$

Summing over $r \in \{0, \dots, R-1\}$ (for any integer $R > 0$) gives:

$$\mathbb{E} \{L[R]\} - \mathbb{E} \{L[0]\} \leq BR + \sum_{r=0}^{R-1} \sum_{m=1}^M \mathbb{E} \{Q_m[r]\} T^* \lambda_m (1 - 1/\rho)$$

Using the fact that $\mathbb{E} \{L[r]\} \geq 0$ and $\mathbb{E} \{L[0]\} = 0$, dividing by R , and rearranging terms gives:

$$\frac{1}{R} \sum_{r=0}^{R-1} \sum_{m=1}^M \lambda_m \mathbb{E} \{Q_m[r]\} \leq \frac{B\rho}{T^*(1-\rho)}$$

Because $T^* \geq 1$, the above bound can be simplified to $B\rho/(1-\rho)$. The above holds for all R , and so the expected queue backlog is $O(1/(1-\rho))$. Further, from [7] we can derive that the following holds with probability 1:

$$\limsup_{R \rightarrow \infty} \frac{1}{R} \sum_{r=0}^{R-1} \sum_{m=1}^M \lambda_m Q_m[r] \leq \frac{B\rho}{T^*(1-\rho)} \quad \square$$

Proof: (Queue Bound for Algorithm 2) Recall that (28) holds for every frame r and all possible $\mathbf{Q}[r]$ for Algorithm 2. Using $\alpha^*[r]$ as an algorithm that makes randomized decisions on frame r that are independent of $\mathbf{Q}[r]$ gives:

$$\frac{\mathbb{E} \left\{ \sum_{m=1}^M Q_m[r] \mu_m(\alpha[r]) | \mathbf{Q}[r] \right\}}{\mathbb{E} \{T(\alpha[r]) | \mathbf{Q}[r]\}} \geq \frac{\sum_{m=1}^M Q_m[r] \mathbb{E} \{ \mu_m(\alpha^*[r]) \}}{\mathbb{E} \{T(\alpha^*[r])\}} \quad (33)$$

where we have removed the conditional expectations on the right-hand-side. Because $(\lambda_m/\rho) \in \Lambda_{\mathcal{A}}$, we know that there

is an algorithm that makes independent and randomized decisions to yield (32). Plugging (32) into the right-hand-side of (33) gives:

$$\frac{\mathbb{E} \left\{ \sum_{m=1}^M Q_m[r] \mu_m(\alpha[r]) |Q[r] \right\}}{\mathbb{E} \{T(\alpha[r]) |Q[r]\}} \geq \sum_{m=1}^M Q_m[r] \lambda_m / \rho$$

Rearranging gives:

$$\sum_{m=1}^M Q_m[r] \mathbb{E} \{ \mu_m(\alpha[r]) |Q[r] \} \geq \frac{1}{\rho} \sum_{m=1}^M Q_m[r] \mathbb{E} \{ \lambda_m T(\alpha[r]) |Q[r] \}$$

Using this in the drift bound of Lemma 3 gives:

$$\Delta[r] \leq B + \sum_{m=1}^M Q_m[r] \mathbb{E} \left\{ \lambda_m T(\alpha[r]) - \frac{\lambda_m}{\rho} T(\alpha[r]) |Q[r] \right\}$$

That is:

$$\begin{aligned} \Delta[r] &\leq B + \sum_{m=1}^M Q_m[r] \lambda_m (1 - 1/\rho) \mathbb{E} \{T(\alpha[r]) |Q[r]\} \\ &\leq B + \sum_{m=1}^M Q_m[r] \lambda_m (1 - 1/\rho) \end{aligned}$$

where we have used the fact that $\mathbb{E} \{T(\alpha[r]) |Q[r]\} \geq 1$, and $1 - 1/\rho \leq 0$. We thus have by the same argument as in the previous proof that for any $R > 0$:

$$\frac{1}{R} \sum_{r=0}^{R-1} \sum_{m=1}^M \lambda_m \mathbb{E} \{Q_m[r]\} \leq \frac{\rho B}{1 - \rho}$$

and with probability 1:

$$\limsup_{R \rightarrow \infty} \frac{1}{R} \sum_{r=0}^{R-1} \sum_{m=1}^M \lambda_m Q_m[r] \leq \frac{\rho B}{1 - \rho}$$

□

APPENDIX E — THE DISJOINT CYCLE THEOREM FOR GENERAL INDEX CODING

The weighted compressed graph $\mathcal{WC}(\mathcal{G})$ was introduced in Section IV for the context of broadcast relay networks, being special cases of index coding problems where each packet has exactly one incoming link and exactly one outgoing link. However, such a graph is also useful for general index coding problems, with general directed bipartite graphs \mathcal{G} .

Consider any such general directed bipartite graph \mathcal{G} with user nodes \mathcal{N} and packet nodes \mathcal{P} . Define $\mathcal{WC}(\mathcal{G})$ similarly: It is a graph on the user nodes \mathcal{N} , a link (i, j) exists if and only if user i has a packet as side information that is wanted by user j , and each link is weighted by P_{ij} , the (integer) number of distinct packets of this type. An example of a graph \mathcal{G} and its weighted compressed graph $\mathcal{WC}(\mathcal{G})$ is given in Fig. 6.

Note that weighted compressed graphs in this case typically include less information than the original graph \mathcal{G} . For example, the existence of a link (a, b) with weight P_{ab} in the graph $\mathcal{WC}(\mathcal{G})$ tells us that node a has P_{ab} distinct packets that are

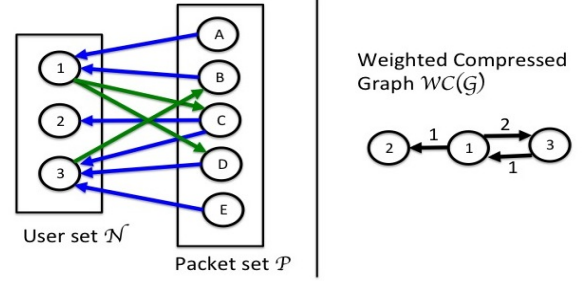


Fig. 6. An example graph \mathcal{G} and its weighted compressed graph $\mathcal{WC}(\mathcal{G})$.

wanted by node b , but does not specify which packets these are, or if these packets also take part in the weight count on other links of $\mathcal{WC}(\mathcal{G})$. Indeed, in the example of Fig. 6, packet C is the packet corresponding to the weight 1 on link $(1, 2)$ in $\mathcal{WC}(\mathcal{G})$, and packets C and D correspond to the weight 2 on link $(1, 3)$ in $\mathcal{WC}(\mathcal{G})$, so that the same packet C is included in the weight for two different links. Note also that, unlike broadcast relay networks, the sum of the weights of $\mathcal{WC}(\mathcal{G})$ is not necessarily equal to the number of packets P in \mathcal{G} .

Lemma 4: The original demand graph \mathcal{G} is acyclic if and only if $\mathcal{WC}(\mathcal{G})$ is acyclic.

Proof: We show that \mathcal{G} has cycles if and only if $\mathcal{WC}(\mathcal{G})$ has cycles. The proof can be understood directly from Fig. 7. Suppose that $\mathcal{WC}(\mathcal{G})$ has a cycle of length K , as shown in Fig. 7a. Relabeling the nodes, this cycle uses nodes $\{1, \dots, K\}$, and has links $\{(1, 2), (2, 3), \dots, (K-1, K), (K, 1)\}$. However, for each link (a, b) of this cycle, there must be a packet $x_{ab} \in \mathcal{P}$ such that $x_{ab} \in \mathcal{H}_a$ (so the graph \mathcal{G} has a directed link from user node a to packet node x_{ab}), and $x_{ab} \in \mathcal{R}_b$ (so the graph \mathcal{G} has a directed link from packet node x_{ab} to user node b). This means that the original graph \mathcal{G} has a cycle, as depicted in Fig. 7b.

Conversely, if the bipartite graph \mathcal{G} has a cycle, it must alternate between user nodes and packet nodes, with a structure as depicted in Fig. 7b. From this, it is clear that $\mathcal{WC}(\mathcal{G})$ also has a cycle. □

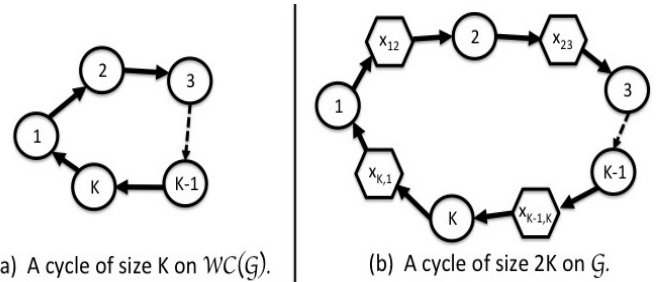


Fig. 7. An illustration for the proof of Lemma 4.

Lemma 4 makes it easier to see whether or not a given graph \mathcal{G} is acyclic. That the graph \mathcal{G} in Fig. 6 has cycles is immediately apparent from its much simpler weighted compressed graph $\mathcal{WC}(\mathcal{G})$, which has a single cycle of size 2 consisting of nodes 1 and 3. As another example, it may not be immediately clear that the graph \mathcal{G} in Fig. 1 is acyclic. However, its graph $\mathcal{WC}(\mathcal{G})$ has only 3 nodes $\mathcal{N} = \{1, 2, 3\}$

and two links (1,3) and (3,2), and from this the acyclic structure is obvious. These compressed graphs are also useful for coding, even when they have cycles, as shown next.

We say that a packet is a *unicast packet* if it has at most one outgoing link (so that it is intended for only one destination user), and a packet is a *multicast packet* if it is intended for more than one destination user. We say that the weighted graph $\mathcal{WC}(\mathcal{G})$ has disjoint cycles if no link participates in more than one cycle. Suppose now that $\mathcal{WC}(\mathcal{G})$ has disjoint cycles, and let K be the number of such cycles. As before, for each $k \in \{1, \dots, K\}$, we let $\mathcal{C}^{(k)}$ represent the set of links for the k th cycle, and let $w_{\min}^{(k)}$ represent the weight of the minimum weight link in $\mathcal{C}^{(k)}$. Note that removing this link from each cycle results in a new graph that is acyclic. This is used in the following theorem.

Theorem 5: Let \mathcal{G} be a demand graph with N nodes and P packets. Suppose that $\mathcal{WC}(\mathcal{G})$ has disjoint cycles, that there are K such cycles, and that all packets of these cycles are distinct and are unicast packets. Then:

$$T_{\min}(\mathcal{G}) = P - \sum_{k=1}^K w_{\min}^{(k)}$$

where P is the number of packets in \mathcal{G} . Furthermore, the minimum clearance time can be achieved by performing cyclic coding $w_{\min}^{(k)}$ times for each cycle $k \in \{1, \dots, K\}$, and then transmitting all the remaining packets without coding.

Proof: For each cycle $k \in \{1, \dots, K\}$, select a link with a link weight equal to the minimum link weight $w_{\min}^{(k)}$ (breaking ties arbitrarily). Let $\mathcal{P}^{(k)}$ be the set of all packets associated with this link. All packets in $\cup_{k=1}^K \mathcal{P}^{(k)}$ are distinct (by assumption), and the total number of these packets is:

$$|\cup_{k=1}^K \mathcal{P}^{(k)}| = \sum_{k=1}^K w_{\min}^{(k)}$$

Now consider the subgraph \mathcal{G}' formed from \mathcal{G} by removing all packet nodes in the set $\cup_{k=1}^K \mathcal{P}^{(k)}$. The number of packets P' in this graph is thus:

$$P' = P - \sum_{k=1}^K w_{\min}^{(k)}$$

Further, we have $T_{\min}(\mathcal{G}') \leq T_{\min}(\mathcal{G})$. However, note that $\mathcal{WC}(\mathcal{G}')$ is the same as $\mathcal{WC}(\mathcal{G})$, with the exception that the min-weight link on each of the K cycles has been removed. Thus, $\mathcal{WC}(\mathcal{G}')$ is acyclic, so that \mathcal{G}' is acyclic, and by Theorem 1 we have $T_{\min}(\mathcal{G}') = P'$. It follows that:

$$T_{\min}(\mathcal{G}) \geq P'$$

Thus, any algorithm for clearing all packets in the graph \mathcal{G} must use at least P' slots. We now show that $T_{\min}(\mathcal{G}) = P'$ by designing a simple cyclic coding scheme to clear all packets in the original graph \mathcal{G} in exactly P' slots. By assumption, all packets that participate in cycles are distinct unicast packets, and hence they only need to be delivered to one destination, as defined by the link of the cycle they are in. This includes all packets in the set $\cup_{k=1}^K \mathcal{P}^{(k)}$. We now use the obvious strategy: For each cycle $k \in \{1, \dots, K\}$, we choose an undelivered

packet $p \in \mathcal{P}^{(k)}$, and use a cyclic coding operation that clears $S^{(k)}$ distinct packets (including packet p) in $S^{(k)} - 1$ slots, where $S^{(k)}$ is the size of cycle k . This can be done because packet p is from the link of cycle k with the fewest packets, and so there are always remaining packets on the other links of the cycle to use in the coding. Once this is done for all cycles and for all packets on the min-weight link of each cycle, we then transmit the remaining packets uncoded. The total number of transmissions is thus equal to P minus the savings of $\sum_{k=1}^K w_{\min}^{(k)}$ from all of the cyclic coding operations. Thus, this takes exactly $P - \sum_{k=1}^K w_{\min}^{(k)} = P'$ slots. \square

Corollary 1: If the demand graph \mathcal{G} has P packets, but only two users (so that $\mathcal{N} = \{1, 2\}$), then:

$$T_{\min}(\mathcal{G}) = P - w_{\min}$$

where $w_{\min} \triangleq \min[P_{12}, P_{21}]$, being the weight of the min-weight link in the graph $\mathcal{WC}(\mathcal{G})$. Further, this minimum clearance time can be achieved by performing cyclic coding over the packets associated with this min weight link, and then transmitting (uncoded) all remaining packets.

Proof: It suffices to show that $\mathcal{WC}(\mathcal{G})$ has disjoint cycles, and that all packets that are part of these cycles are distinct unicast packets. The graph $\mathcal{WC}(\mathcal{G})$ has only 2 nodes and hence at most one cycle, so it clearly satisfies the disjoint cycle criterion. All packets of the cycle are clearly distinct. Indeed, two packets on different links of the cycle must have one packet on link (1,2) and the other on link (2,1), and hence the first is desired by user 2, while the second is *not* desired by user 2 (so they cannot be the same packet). It remains to show that all packets in the cycle are unicast packets. If there are no cycles, we are done. If there is a cycle, this involves link (1,2) and link (2,1). Any packet associated with the link (1,2) must already be in the set of packets that node 1 has, and so this packet only requires transmission to node 2 (not to node 1). Thus, this packet must be a unicast packet. Similarly, any packet associated with link (2,1) must be a unicast packet. \square

APPENDIX F — OPTIMALITY OF CYCLIC CODING FOR RELAY NETWORKS WITH $N = 3$

Consider now the special case of broadcast relay networks, where each packet node of the graph \mathcal{G} has exactly one incoming link and one outgoing link. There are $N = 3$ users and P packets (where P is a positive integer). This can be exactly represented by the weighted compressed graph $\mathcal{WC}(\mathcal{G})$ with link weights P_{ij} , where $P = \sum_{i=1}^N \sum_{j=1}^N P_{ij}$. We want to show that minimum clearance time can be achieved by cyclic coding (using either direct transmission, 2-cycle code actions, or 3-cycle code actions). If the 3-node graph $\mathcal{WC}(\mathcal{G})$ has disjoint cycles, we are done (recall Theorem 4).

Consider now the general case with possibly non-disjoint cycles, as shown in Fig. 8. To represent the general case, we allow link weights P_{ab} to possibly be 0 (a weight of 0 is equivalent to the absence of a link). First define \min_{12} , \min_{23} , \min_{31} as the weight of the min-weight link for each of the

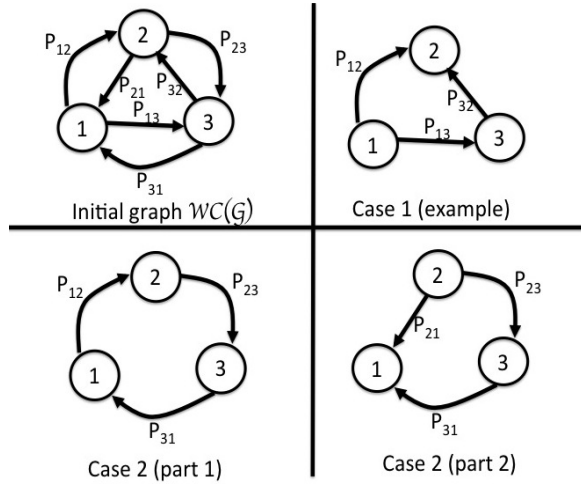


Fig. 8. An illustration of the general broadcast relay graph $\mathcal{WC}(\mathcal{G})$ with $N = 3$ users, and the two cases required for the proof.

three possible 2-cycles:

$$\begin{aligned} \min_{12} &= \min[P_{12}, P_{21}] \\ \min_{23} &= \min[P_{23}, P_{32}] \\ \min_{31} &= \min[P_{31}, P_{13}] \end{aligned}$$

Now prune the graph $\mathcal{WC}(\mathcal{G})$ by removing the min-weight link for each 2-cycle. This results in a graph $\mathcal{WC}(\mathcal{G}')$ with 3 nodes and (at most) 3 links, and with a total number of packets equal to $P - \min_{12} - \min_{23} - \min_{31}$, as shown in Cases 1 and 2 in the figure. We have two cases.

Case 1: The resulting graph $\mathcal{WC}(\mathcal{G}')$ is acyclic. An example of this case is shown as case 1 in Fig. 8. Thus, we know $T_{\min}(\mathcal{G}) \geq T_{\min}(\mathcal{G}') = P - \min_{12} - \min_{23} - \min_{31}$. However, it is easy to see this clearance time bound can be achieved by using 2-cycle code actions on each of the three 2-cycles, and then transmitting the remaining packets uncoded.

Case 2: The resulting graph $\mathcal{WC}(\mathcal{G}')$ consists of a single 3-cycle. The cycle must either be clockwise or counter-clockwise. Without loss of generality, assume clockwise (see Fig. 8, case 2 part 1). Note that:

$$P_{12} \geq P_{21}, P_{23} \geq P_{32}, P_{31} \geq P_{13} \quad (34)$$

This is because we have formed $\mathcal{WC}(\mathcal{G}')$ by removing the min-weight link on each of the three 2-cycles.

Let $z = \min[P_{12} - P_{21}, P_{23} - P_{32}, P_{31} - P_{13}]$, so that z is a non-negative integer. Without loss of generality, assume the min value for z is achieved by link $(1, 2)$, so that $z = P_{12} - P_{21}$. To $\mathcal{WC}(\mathcal{G}')$, add back the link $(2, 1)$ (with weight P_{21}), and remove the link $(1, 2)$, to yield a graph $\mathcal{WC}(\mathcal{G}'')$ that is an acyclic subgraph of the original graph $\mathcal{WC}(\mathcal{G})$, as shown in case 2 part 2 of Fig. 8. The acyclic subgraph $\mathcal{WC}(\mathcal{G}'')$ contains exactly $P_{21} + P_{23} + P_{31}$ packets. Thus, the minimum clearance time of the original graph $\mathcal{WC}(\mathcal{G})$ is at least $P_{21} + P_{23} + P_{31}$. However, this can easily be achieved. Do the following: Perform 2-cycle code actions on each of the three 2-cycles of the original graph $\mathcal{WC}(\mathcal{G})$, to remove a number of packets on each cycle equal to the min weight link of that cycle. This removes $2P_{21} + 2P_{32} + 2P_{13}$ packets in $P_{21} + P_{32} + P_{13}$ slots

(recall the three min weights are given by (34)). Then perform the 3-cycle code action to remove $3z$ packets in $2z$ slots. Then perform direct transmission to remove the remaining packets, being a total of $x = P - 2P_{21} - 2P_{32} - 2P_{13} - 3z$. The total number of transmissions is:

$$\begin{aligned} &P_{21} + P_{32} + P_{13} + 2z + x \\ &= P_{21} + P_{32} + P_{13} + 2z \\ &\quad + P - 2P_{21} - 2P_{32} - 2P_{13} - 3z \\ &= P - P_{21} - P_{32} - P_{13} - z \\ &= P_{12} + P_{23} + P_{31} - z \\ &= P_{12} + P_{23} + P_{31} - (P_{12} - P_{21}) \\ &= P_{21} + P_{23} + P_{31} \end{aligned}$$

and so the above scheme is optimal.

REFERENCES

- [1] Y. Wu, P. A. Chou, and S-Y Kung. Information exchange in wireless networks with network coding and physical-layer broadcast. *Conference on Information Sciences and Systems, Johns Hopkins University*, March 2005.
- [2] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Médard. The importance of being opportunistic: practical network coding for wireless environments. *Proc. 43rd Annual Allerton Conf. on Communication, Control, and Computing*, Oct. 2005.
- [3] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: Practical wireless network coding. *Proc. ACM SIGCOMM*, 2006.
- [4] Y. Birk and T. Kol. Informed-source coding-on-demand (iscod) over broadcast channels. *Proc. IEEE INFOCOM*, 1998.
- [5] Y. Birk and T. Kol. Coding-on-demand by an informed source (iscod) for efficient broadcast of different supplemental data to caching clients. *IEEE Transactions on Information Theory*, vol. 52, pp. 2825-2830, 2006.
- [6] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol. Index coding with side information. *IEEE Transactions on Information Theory*, vol. 57, no. 3, March 2011.
- [7] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.
- [8] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 466-478, March 1993.
- [9] E. Lubetzky and U. Stav. Nonlinear index coding outperforming the linear optimum. *IEEE Transactions on Information Theory*, vol. 55, no. 8, pp. 3544-3551, Aug. 2009.
- [10] M. J. Neely. Queue stability and probability 1 convergence via lyapunov optimization. *Arxiv Technical Report*, Oct. 2010.